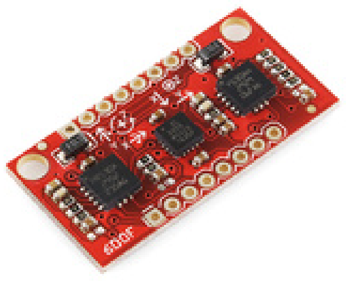
Guide to gyro and accelerometer with Arduino including Kalman filtering

Apr 09, 2011, 09:25 pm Last Edit: Apr 17, 2014, 05:04 pm by Lauszus Reason: 1

Hallo everybody
I recently bought this analog 6DOF (six degrees of freedom) IMU board (http://www.sparkfun.com/products/10010) from watterott.com. It uses three gyros and three accelerometers to calculate angles in three dimensions.



I looked a while for some code online and how to connect with them. After many hours of research I succeeded of making af precise measurement of angles in two directions. I decided to write a short guide for fellow electronic enthusiasts.
The main purpose of this guide is to teach others how to get some useful data from their IMU or just a gyro or accelerometer. The code for Arduino can be found at github: https://github.com/TKJElectronics/Example-Sketch-for-IMU-including-Kalman-filter. It should be pretty easy to implement my code to your own sensor. I will not describe all the details about the theory behind, instead you can look at the sources for more info.

Before you begin you have to connect the IMU as follows:

Acc_Gyro		Arduino
3.3V	<-->	3.3V
GND	<-->	GND
Gx4 X	<-->	AN0
Gx4 Y	<-->	AN1
Gx4 Z	<-->	AN2
Acc X	<-->	AN3
Acc Y	<-->	AN4
Acc Z	<-->	AN5

Also connect 3.3V to the AREF pin on the Arduino for more accuracy.
It is VERY important that you do not connect the sensor to 5V - this will destroy the sensor.

Now your are ready for reading some data from the sensor.

To communicate with the sensor is straightforward:
The gyro measures degrees per second (^{0}/s) while the accelerometer measures acceleration (g'a) in three dimensions. Both outputs the measurements as a analog signal.
To get these translated into degrees you have to do some coding:

The gyro
First you have to translate quids (a number from 0-1023) into something useful (this is for a ADC with a 10 bit resolution, for example this should be 4095 (2^12-1=4095) for 12 bit ADC). To do this I just use this simple equation:
gyroRate = (gyroAdc-gyroZero)/sensitivity - where gyroAdc are the readed value from our sensor, gyroZero is the value when it is stationary (this is done in the code - look in the "Setup" section) while sensitivity is the sensitivity found in the datasheet, but translated into quids.

If you look in the two gyros datasheets (http://www.sparkfun.com/datasheets/Sensors/IMU/lpr530al.pdf and http://www.sparkfun.com/datasheets/Sensors/IMU/LY530ALH.pdf) you will see that the sensitivity is 3.33mV/^{0}/s for the 4xOUT. To translate these into quids is pretty easy: sensitivity/3.3*1023.
So in this example I get:
0.00333/3.3*1023=1.0323.

NB: to translate mV to V simple just divide by one thousand.

The final equation will look like this:
gyroRate = (gyroAdc-gryoZero)/1.0323

The result will come out as degrees per second (^{0}/s). To translate this into degrees you have to know the exact time since the last loop. Fortunately, the Arduino got a simple command to do so: millis(). By using that, one can calculate the time difference (delta time) and thereby calculate the angle of the gyro. The final equation will look like this:
gyroAngle += gyroRate*dtime/1000

Unfortunately, the gyro drifts over time. That means it can not be trusted for a longer timespan, but it is very precise for a short time. This is when the accelerometer comes in handy. It does not have any drift, but it is too unstable for shorter timespan. I will describe how to combine these measurements in a while, but first I will describe how to translate the readings from the accelerometer into something useful.

The accelerometer

The accelerometer measures the acceleration (g's) in three dimensions. To translate the analog readings into degrees you simply need to read the axis and to subtract the zero offset like so:
accVal = accAdc-accZero

Where accAdc is the analog reading and accZero is the value when it reads 0g - this is calculated in the start of the code, look in the "Setup" section. The zero value can also be found in the datasheet: http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf. You will see that the zero voltage at 0g is approximately 1.5V, to translate this into quids, you again have to use this equation: zeroVoltage/3.3*1023.
So in this example I get:
1.5/3.3*1023=465.

You can then calculate the pitch and roll using the following equations:
pitch = atan2(accYval, accZval)+PI
roll = atan2(accXval, accZval)+PI

Atan2 has a output range from -? to ? (see http://en.wikipedia.org/wiki/Atan2), I simply add ?, so the range it converted to 0 to 2?.
To convert it from radians to degrees we simply multiply the result by 57.295779513082320876798154814105 - this is predefined in the Arduino IDE as RAD_TO_DEG.

Kalman filter

As I explained earlier the gyro is very precise, but tend to drift. The accelerometer is a bit unstable, but does not drift. You can calculate the precise angle by using something called a Kalman filter. A detailed guide on how it's implemented can be found at my blog:
http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/.

If you want to use something a bit more simple, you can use what's called a Complementary Filter. It is pretty easy to understand and the math is much simpler, because it only works in one step.
For example the equation could look like this:
angle = 0.98 *(angle+gyro*dt) + 0.02*acc - you can fine tune the numbers to whatever you like. Just remember that the sum must be 1.
For me the result from the Complementary Filter was very close (or almost the same) as the one calculated by the Kalman filter.

You have now learned (hopefully) how to get analog data from IMU and translate it to something useful. I have attached my own code for my 6DOF IMU (http://www.sparkfun.com/products/10010), but with some slightly modification, I am pretty sure that it is possible to use it with any analog gyro/accelerometer.

If you have any question, fell free to post a comment below.

Sources:
http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1284738418
http://www.x-firm.com/?page_id=148
http://web.mit.edu/first/segway/

Update
I have just finished a Processing code which prints out data from the Arduino on a nice graph. As you can see in the video below the filtering is quit effective. The light blue line is the accelerometer, the purple line is the gyro, the black line is the angle calculated by the Complementary Filter, and the red line is the angle calculated by the Kalman filter. As you might see the Kalman filter is just a bit more precise (i know it is difficult to see in the video) than the Complementary Filter, especially when I shake it.
The code can be found at github: https://github.com/TKJElectronics/Example-Sketch-for-IMU-including-Kalman-filter/tree/master/Graph.
It is also possible to see the data from the y-axis. Just uncomment *drawAxisY()*; in the code.

http://www.youtube.com/watch?v=sQC73LKnJck

My Balancing robot
Below is a video of my balancing robot. It uses the same IMU and algorithm as described in the post above.
The source code can be found at the following link:
https://github.com/TKJElectronics/BalancingRobotArduino
http://www.youtube.com/watch?v=N28C_JqVhGU

Kickstarter
I have just released my balancing robot on Kickstarter:
http://www.kickstarter.com/projects/tkjelectronics/balanduino-balancing-robot-kit.
Please consider backing the project.
http://www.youtube.com/watch?v=_kQniPbg9zc

Update
I have also provided code for several other IMUs including a 9DOF which allows you to estimate yaw as well.
All can be found in the Github repository: https://github.com/TKJElectronics/Example-Sketch-for-IMU-including-Kalman-filter.

Hey Lauszus!

First off thanks for posting, this looks great and could be exactly what i need!

basically i am creating a ball full of different sensors which will then act as a synthasizer.this is my final year project and i am stuck here with only a few weeks to go. I have other sensors but what i really want is to be able to read if the ball is being tilted forward or back,left and right and by how much so i can modulate sound accordingly.(i am not trying to measure the acceleration,just the amount it is tilted)

i have been searching all the different possibilities with gyroscopes and accelerometers but can't seem to find clear cut examples of what i want. It looks like your code here is what i need? If i was to buy the 6dof is it as simple as using the code you linked? and then i would be able to get proper tilt values?

I am only a beginner with arduino and just know the basics of reading sensors but i'm finding the task of measuring tilt a bit deep for me.

your help or advice would be greatly appreciated!

(a final year college student freaking out lol) Billy.

#35

May 19, 2011, 12:50 am **Last Edit: May 19, 2011, 02:56 am by Lauszus Reason: 1**

Yes it is yaw (the rotation around the z-axis). The reason why you can not measure yaw with a accelerometer is because the gravitational force from earth does not change when you rotate the IMU horizontal. In theory you will not measure any difference at all, in pratical however you see the acceleration from the movement. I use the z-value from the accelerometer together with pitch and roll to calculate the force vector (this is theoretical 1, but in practice there is a small difference). I also use the z-value from the accelerometer to know when the IMU is tilted more than 90 degrees to both side, so I can get 360 degree resolution instead of only 180

- Lauszus
hey Lauszus,

I got my sensor (nice and fast too,thanks for the germany tip!)

I have it all connected and it seems to be working. One thing i notice though is that tilting forward on the y-axis gives me values up to 60 but backwards on the y-axis i only get as high as 20. on the x-axis then i an even 40 on each side.

I notice you wrote to add in a piece of code to make it read full 360, is that why mine isn't reading right?

i tried putting in that code after

```
"  R = sqrt(pow(accXval,2)+pow(accYval,2)+pow(accZval,2));//the force vector
  accXangle = acos(accXval/R)*RAD_TO_DEG-90;
  accYangle = acos(accYval/R)*RAD_TO_DEG-90;"
```

But didnt change anything.

also i'm not connecting the z-axis because i don't need it. I assumed it wouldnt make a difference.

Just wanted to say thanks again for the code. Looks like my project will become what i dreamed

😃

I'll post a video up here when its done